UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

Comic Style Rendering

Treball Final de Grau

Grau en Disseny i Desenvolupament de Videojocs

Cognoms: Izquierdo Cunill Nom: Miquel

Pla: 2014

Director: Pillosu Gonzalez, Ricardo Javier

Índex

Resum

Aquest treball parteix de la idea de voler recrear l'estil de la il·lustració 2D, en concret la del còmic, per a poder-lo fer servir en el desenvolupament d'un videojoc.

En aquest treball s'analitzarà primer l'estil d'un artista en concret, mirant les tècniques que utilitza i com es podrien fer servir per a videojocs, per a més endavant aplicar-les en una demo amb Unity, per a poder-nos per una idea de com podria quedar un videojoc amb aquest estil.

El treball consisteix en el desenvolupament i explicació de totes les tècniques (shaders, efectes de postprocessat, ...) utilitzades en la demo final.

Paraules clau

Programació gràfica, Unity, shaders, render, videojoc, comic, CelShading

Enllaços

Github: https://github.com/mizquierdo97/ComicRendering

Build: https://github.com/mizquierdo97/ComicRendering/releases/tag/0.5

WebGL: https://mizquierdo97.github.io/ComicRendering/Web/ (Obrir amb GoogleChrome)

Índex de taules

Índex de figures

Glossari

Unity: Software usat per al desenvolupament de videojocs.

Render Pipeline: Procés de convertir gràfics tridimensionals en una imatge 2D mitjançant shaders.

Shader: Programa que funciona en una targeta gràfica que realitza càlculs com la il·luminació, color… Per a aplicar-los a una geometría.

FPS: De l'anglès Frames Per Second. Fotogrames per segon als que s'executa un videojoc.

Outline: Contorn dels objectes. Normalment usat en il·lustració.

Asset: Recursos fets servir en un videojoc. Models 3D, imatges, audios...

Geometria: Models 3D tant de personatges escenaris.

Textura: Imatges usades per a donar color o altres atributs a la geometria.

Normals: Vector perpendicular a la geometria.

1. Introducció

1.1 Motivació

Des d'un principi sabia el tema del meu Treball Final de Grau, estaria relacionat amb la programació gràfica, ja que és un dels camps dels videojocs que trobo molt interessant, ja que combina tant una part de programació, amb els Shaders i la pipeline de Render, com part artística, ja que al final el resultat ha de quedar visualment interessant.

Inicialment la idea que vaig tenir per al TFG era de fer un editor de Shaders mitjançant nodes, a l'igual que tenen alguns motors de videojocs com Unity o Unreal i que permet programar i editar els shaders de manera visual.

Finalment, a causa de començar les pràctiques d'empresa a Arhat Games se'm va proposar de canviar el tema del treball a algo que pogués estar més relacionat amb l'empresa.

Així doncs, se'm va proposar d'intentar recrear l'estil artístic del còmic, concretament l'estil del Josan Gonzalez[1], de manera que es pogués aconseguir un resultat semblant mitjançant un motor de videojoc. Després de documentar-me una mica sobre com es podria aconseguir el resultat desitjat i de quins mètodes hauria de fer servir, vaig optar per tirar endavant amb aquest tema, ja que desde el moment en el qual me'l van presentar vaig creure que podria ser un treball interessant i que em suposaria més reptes dels que podia semblar des d'un principi.

F1 - Il·lustració de Josan Gonzalez

[1] Il·lustrador https://www.artstation.com/josan

1.2 Formulació del problema

Actualment en la majoria de videojocs, s'intenta aconseguir un apartat gràfic característic, ja sigui fent que s'acosti el màxim possible a la realitat o optant per un estil diferent i únic.

A causa que gran part dels videojocs que surten al mercat opten per un apartat visual realista, la majoria de recursos van enfocats a què aquest estil sigui cada cop millor, més òptim i més fàcil de fer servir. Per altra banda, altres estils artístics com podrien ser el del còmic, pixel-art o cartoon, no són tan comuns. Això fa que hi hagi menys documentació per a recrear aquests estils, que els motors no estiguin tan preparats per a treballar les diferents necessitats que requereixen i per tant, fa que hi hagi menys videojocs amb aquests estils.

Per aquestes raons, la idea d'aquest treball, és intentar recrear un d'aquests estils, concretament l'estil del còmic. A part, l'objectiu d'aquest treball també consistirà en documentar el procés que s'ha seguit per a obtenir el resultat final i els mètodes utilitzats, per a més tard, crear una documentació que faciliti a aquella gent que vulgui recrear un estil semblant.

1.3 Objectius generals del TFG

L'objectiu principal d'aquest treball és el d'aconseguir recrear l'estil dels còmics utilitzant una metodologia de treball 3D amb Unity. A més, documentar tot el procés que s'ha seguit per a aconseguir el resultat final.

Estil Gràfic

L'objectiu principal i més important d'aquest projecte, és el d'aconseguir recrear l'estil artístic del Josan Gonzalez. El propòsit, és el de crear una metodología que sigui aplicable per a més tard, poder crear un videojoc. Aquesta metodologia, ha de ser aplicable per a un videojoc en 3D pero que el resultat gràfic final sembli una il·lustració 2D.

Pipeline de Unity

Com que Unity és un dels motors de videojocs més utilitzats, és interessant aprendre com funciona internament la pipeline de render. Aquests coneixements, també quedaran inclosos en la documentació, ja que actualment hi ha poca documentació respecte aquest tema.

Documentació

Un altre objectiu, és el de documentar tot el procés que s'ha seguit per a aconseguir el resultat final, tant els mètodes que s'han utilitzat com els passos necessaris per a fer-los servir en Unity. La idea d'aquesta documentació és la de que altres persones puguin fer servir els coneixements apresos per a aplicar-los en un altre projecte de manera simple.

Mètodes usats en la industria

Per últim, es vol aprendre com es fan servir els mètodes necessaris per a aquest treball, en la indústria dels videojocs. Com per exemple el outline per a la geometria o l'efecte del Glow. Comparar quins son els diferents mètodes per a aconseguir el mateix resultat, comparar pros i contres i veure perquè es fan servir uns mètodes i no uns altres.

1.4 Objectius específics del TFG

Els objectius específics marcats per a aquest treball, els he dividit en funció de les característiques que haurà de tenir el resultat final.

Outline

Per tal de que el resultat final sembli una il·lustració de còmic, una de les característiques més importants que ha de tenir és la del traç del dibuix. Per a aconseguir que aquest tret quedi de la manera desitjada, s'haurà de fer servir varies tècniques, com filtres de post processat, utilització de mapes de normals o textures.

Colors i il·luminació

Un dels trets més importants de les il·lustracions de còmic en general, i de l'estil que s'ha escollit en concret, és la utilització del color i de la il·luminació. Ja que fa servir uns colors molt plans amb una il·luminació i ombres molt subtils.

Efectes i partícules

Un altre aspecte important a tenir en compte, serà les partícules i diferents efectes visuals que requereixi l'estil. Ja que s'hauran d'adaptar els efectes i partícules convencionals a l'estil gràfic.

Animació

L'animació dels personatges, pot ser un dels aspectes més difícils d'aconseguir. Ja que per una part, no hi ha masses exemples de com seria l'animació en l'estil gráfic que és vol recrear, a més que la animació 3D i l'animació 2D és realitzen fent servir uns mètodes molt diferents. Això pot provocar que conseguir una animació en 3D pero que doni la sensació de que sigui animació tradicional en 2D pugui resultar molt difícil.

Rendiment

Per últim, s'haurà d'aconseguir que aquest projecte funcioni a un mínim de 30 FPS, per tal de que sigui funcional per a utilitzar-ho per a un videojoc.

1.5 Abast del projecte

L'abast d'aquest projecte es divideix en dues parts:

La primera consta del projecte de Unity, que inclou tota la programació dels shaders, la pipeline de render i tots els scripts necessaris, així com les escenes i els assets creats. Aquesta primera part anirà dirigira a la empresa Arhat Games, de manera que es puguin beneficiar de la tecnologia desenvolupada per al futur desenvolupament de un posible videojoc.

La segona part del projecte consistirà en tota la documentació generada al finalitzar el projecte. Aquesta documentació serà pública de manera que pugui ser utilitzada per a qualsevol persona per al desenvolupament d'altres videojocs o aplicacions.

2. Estat de l'art

Actualment, només existeixen uns quants videojocs que s'acostin a l'estil artístic al que volem arribar. Tot i això tots els mètodes que utilitzen son bastant comuns en la indústria i es fan servir de manera regular.

Abans d'entrar a veure els videojocs que més s'acosten a l'estil al que volem arribar, caldrà analitzar els quins son alguns dels mètodes que es fan servir a la indústria i quin s'adapta més a les nostres necessitats per tal d'aconseguir el resultat desitjat.

Outline

Una de les característiques més importants per a que el resultat final sembli un còmic, és el outline, ja que està molt associat a la il·lustració 2D i sobretot al còmic.

Aquest mètode s'utilitza communment als videojocs, com a part de l'estil artístic o simplement per a petites característiques del videojoc, com per exemple mostrar objectes importants. Existeixen varis mètodes per a aconseguir aquest efecte depenent de quin ús se li vulgui donar.

Outline per mesh

El mètode més simple per a crear un outline consisteix en duplicar la mesh del objecte al que se li vol aplicar l'efecte, fer-la lleugerament més gran i pintant-la per darrera del objecte, de manera que doni l'efecte de que la Mesh te outline, pero realment sigui un duplicat d'aquesta mesh.

Aquest efecte dona bastants bons resultats quan el que es vol fer és aplicar outline a objectes específics i que tinguin una geometria bastant simple. En canvi si el que es vol és aplicar aquest mètode a tots els objectes d'un videojoc, com és en el meu cas, s'haurien de renderitzar el doble de polígons fent que es ralentitzi bastant el videojoc. A més, amb objectes amb ina geometria més complexa, com podria ser un personatge, comencen a aparèixer errors gràfics. Per a aquests motius, es va descartar aquest mètode ja que no s'adapta a les necessitats del projecte.

F2 - Exemple de outline generat a partir de la mesh

Outline per Textures

El segon mètode, és el de aplicar l'outline directament en la textura de l'objecte que es vol renderitzar. Aquest mètode s'ha fet servir en nombroses ocasions, per el va popularitzar el videojoc **Borderlands.** Simplement consisteix en que en el moment de crear la textura de l'objecte, es pinten les línies que es volen aplicar directament en aquesta textura. Tot i que aquest mètode te certes avantatges ja que permet controlar perfectament com i a on hi haurà línies a més de que no suposa cap cost de computació adicional. Tot i això també comporta certs problemes. El primer és que aquest mètode no permet afegir outline a la geometria tal com si que es podia amb el primer mètode. El segon és que al dependre d'una textura, el outline pot perdre qualitat o distorsionar-se depenent de la proximitat de la càmera. Per això la millor opció és la de combinar aquest mètode amb el mètode per postprocessat que veurem a continuació, per tal d'obtenir una major flexibilitat a la hora de treballar amb l'outline.

F3 - Exemple de outline generat a partir de la textura

Outline per Postprocessat

Per últim, aquest mètode consisteix en, una vegada tens la escena renderitzada sense cap efecte de outline, aplicar uns filtres de postprocessat a la imatge obtinguda per tal d'obtenir aquest efecte. El mètode més utilitzat és el de detecció de cantonades (Edge Detection), on a partir d'una imatge, en aquest cas les obtingudes al renderitzar la escena, es pot obtenir els canvis abruptes de color que serán on es pintarà finalment el outline.

Per tal d'aconseguir una major precisió a la hora de detectar cantonades d'una imatge es poden fer servir diferents buffers de l'escena que s'ha renderitzat. Per exemple, amb el buffer de Normals, podem detectar canvis en l'orientació de triangles, de

manera que sabrem que allà hi ha una cantonada. O amb el buffer de Profunditat, podem detectar si dos objectes estan a profunditats diferents i per tant s'hauria d'aplicar el outline.

Aquest mètode és el que més s'adapta a les necessitats del projecte, ja que permet un gran control a la hora de crear el outline. Tot i això s'haurà de complementar amb el outline per textura, ja que en objectes més complexos, és possible que provoqui errors gràfics.

F4 - Exemple de outline generat a partir d'efectes de postprocessat

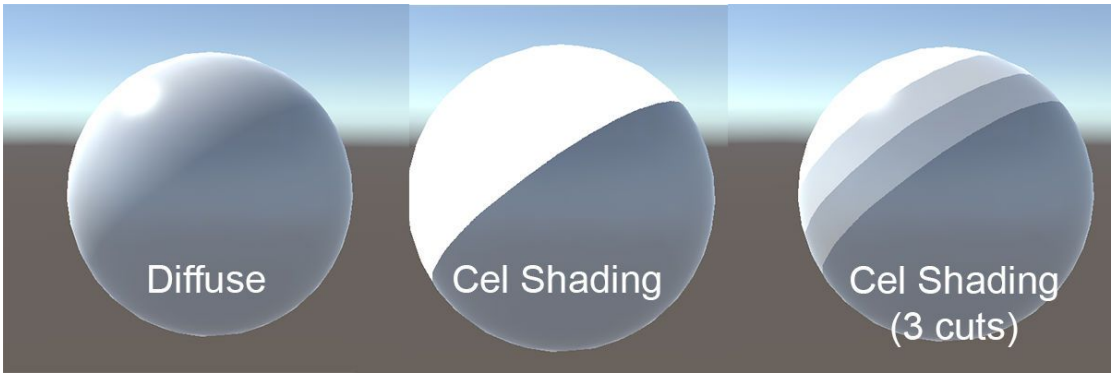
Cel Shading

El cel shading, és un tipus de renderització no-fotorrealistic que intenta simular el dibuix a mà. Es fa servir communment per a videojocs, gràcies a l'estil Cartoon que crea i per el seu poc cost computacional.

Aquesta tècnica té moltes variacions, depenent del resultat final que es vulgui aconseguir, pero totes comparteixen un aspecte en comú. Tots ells fan ús de colors plans amb diferents nivells d'intensitat de les ombres i les il·luminacions per tal que semblin dibuixos.

Aquest mètode es sol fer servir per a videojocs que no volen aconseguir un estil artístic realista, com per exemple The Legend of Zelda: Breath of the Wild o en videojocs per a dispositius mòbils.

Normalment aquest mètodes es complementa amb la utilització de outline.



F5 - Il·luminació cel shading

Glow

L'efecte de Glow o Bloom s'utilitza per a fer que els objectes sembli que tinguin llum pròpia, ja que crea un difuminat al voltant d'aquest.

Actualment, la majoria de videojocs, el fan servir d'una manera o una altre. Això fa que existeixi bastanta documentació dels mètodes que s'han d'aplicar per a obtenir el resultat desitjat.

Per a aconseguir aquest efecte, el mètode més comú consisteix en a partir de la imatge obtinguda al renderitzar l'escena, se li aplica un filtre de postprocessat que difumina la imatge, l'hi augmenta la il·luminació i finalment la combina amb la imatge original.

Aquest mètode es pot aplicar tant per a tota la escena, com per a objectes per separat.

En el nostre cas, l'efecte de glow, ens servirà per donar més detall a les partícules de la escena.

Animació Tradicional

Per últim, per tal d'aconseguir que el resultat final doni l'efecte de ser il·lustració 2D, caldrà adaptar l'animació dels personatges. Actualment hi ha pocs videojocs que

adaptin l'animació per a que sembli animació tradicional en 2D. El cas que s'assembla al que es vol aconseguir, és el del videojoc Guilty Gear X, de l'estudi Arc System Works. El mètode que feien servir en aquest estudi per a animar personatges consisteix en realitzar el model 3D del personatge i animar-lo sense interpolacions als diferents Frames, de manera que el moviment final sigui menys suau. A més en animacions més ràpides i dinàmiques, apliquen una deformació de la mesh perquè doni més sensació de moviment.

Tot i que aquesta tecnica podria quedar molt bé amb l'estil artístic triat, pot comportar varies dificultats. La primera és que els videojocs que la fan servir, son jocs de lluita amb moviment lateral, lo que donava molt control en el resultat final, ja que et permet saber en tot moment on estarà la camara. En el nostre cas, al no tenir una camara fixa, podria complicar molt el procés de creació de les animacions. A per a aconseguir el resultat desitjat es requeriria d'una gran càrrega de treball per part d'art, i és possible que no valgui la pena dedicar tants recursos per a aquesta característica.

F6 - Exemple de deformació de la mesh per a la animació del videojoc Guilty Gear Xrv

2.1 Estudi de Mercat

Aquest treball, no es basa directament en fer un videojoc, si no en crear les eines i tècniques necessàries per a la seva creació. Per tant, té sentit no només analitzar els videojocs amb un estil semblant al que aconseguiriem nosaltres, si no també les eines que hi ha disponibles per a crear-ho.

Eines:

En referencia a les eines, ens hem de fixar amb les opcions que ens proporcionen els diferents motors de videojocs. Actualment no hi ha cap motor que de manera predeterminada, ens ofereixi les eines necessaries per a la creació d'aquest estil gràfic. Però si que ofereixen diverses possibilitats per al desenvolupament.

Ens centrarem en les característiques de Unity i Unreal Engine 4, ja que son els motors més comuns i més accessibles per al desenvolupament de videojocs.

Unreal Engine 4:

Unreal Engine, és un motor enfocat a la creació de videojocs fotorealistes. Això fa que la pipeline de render sigui menys flexible i no et permeti fer certes coses. A més la programació de shaders, es fa de manera visual mitjançant nodes, fet que provoca que en ocasions estigui més limitat i sigui més tedios treballar amb aquest mètode en comparació a una programació de shaders desde codi.

Per contra, els aspectes de la il·luminació, efectes com el glow o animació, estan millor implementats que en Unity. Ja que per exemple l'efecte de Glow ve per defecte en el motor.

Unreal Engine 4, és un motor principalment enfocat al desenvolupament de videojocs per a estudis profesionals, no tant per a desenvolupadors indies. Això fa que no només la quantitat d'assets per a Unreal sigui menor que en Unity, si no que la documentació també sigui menor. Per tant en cas de trobar una limitació amb el motor seria més costos trobar informació per a solucionar-la.

Unity:

Per altre banda, Unity és un motor enfocat a videojocs per a mòbils o videojocs indie. Aquest videojocs, s'allunyen més del fotorrealisme, enfocant-se més a estils cartoon. Això fa que la pipeline de render sigui més flexible, ja que s'ha de poder adaptar a estils molt diferents. És per això que Unity et dóna l'opció de personalitzar tota la pipeline de render del motor de manera que s'adapti a les necessitats que tingui el teu videojoc. A més, la programació dels shaders es fa mitjançant codi, permetent una major complexitat. Per altre banda, la il·luminació, animació i efectes son menys flexibles que en Unreal Engine 4.

Tot i axió la Asset Store de Unity compta amb una gran quantitat d'assets, tan gratuïts com de pagament, que ajuda a solucionar les carències del motor.

En referencia al mercat d'assets de Unity, existeixen alguns productes que es podrien fer servir per a crear un videojoc amb l'estil d'aquest projecte. El principal problema de treballar amb aquests assets, és que es venen per separat. És a dir, hi ha un asset per a generar Glow, un altre per a generar el outline... Això pot provocar que facin servir mètodes no complementaris, i que no es puguin utilitzar junts, o no doni el resultat esperat.

Videojocs:

Com ja s'ha comentat a l'apartat anterior, hi ha un gran número de videojocs que fan servir alguna de les tècniques comentades, però hi ha pocs casos en els que el resultat s'acosti al que es vol aconseguir.

El videojoc més semblant en tots els aspectes, és Sable. Que tot i que encara no ha sortit al mercat, ja s'ha pogut veure alguns detalls de l'apartat visual.

F7 - Fotograma del videojoc Sable

Sable, s'acosta molt al resultat que es voldria aconseguir, sobretot en l'apartat de el outline i dels colors, ja que intenta recrear un estil molt semblant al d'aquest treball. Per altre banda, la il·luminació i les animacions dels personatges s'allunyen més del resultat desitjat.

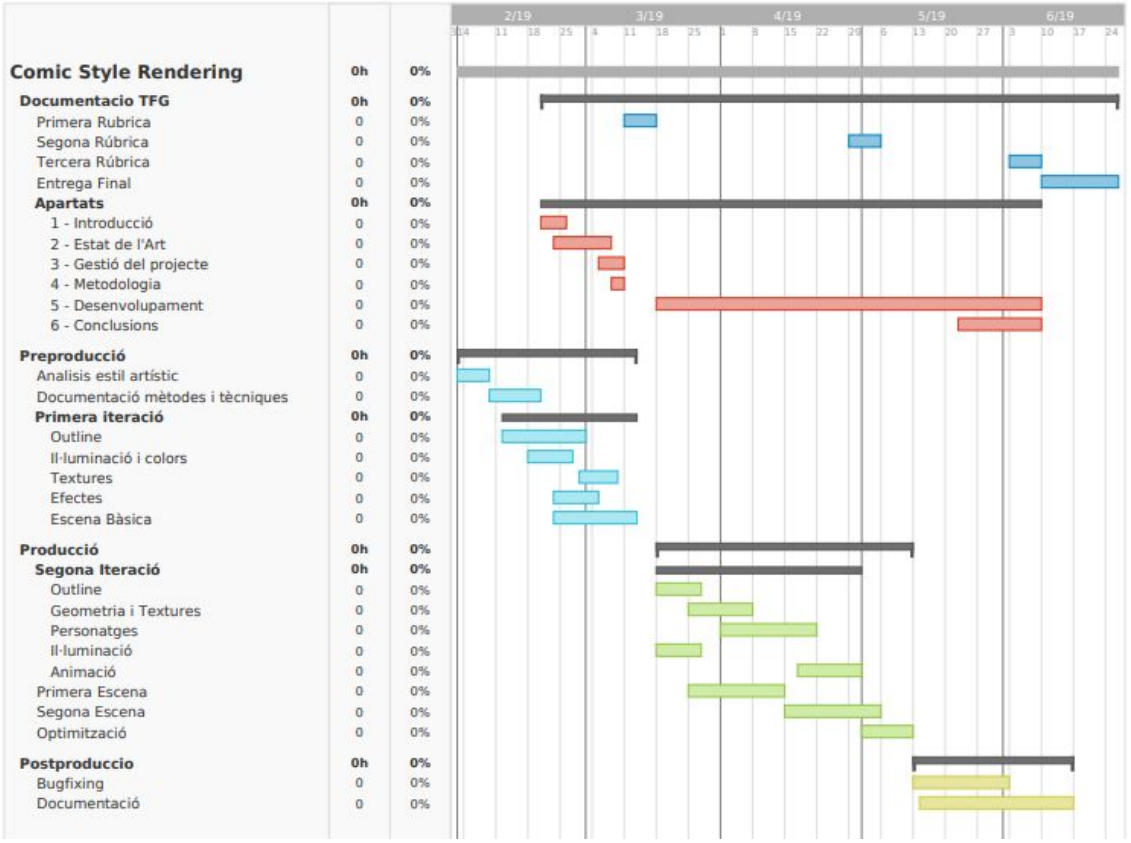
També existeixen alguns videojocs que tot i que tenen un apartat visual diferent, han intentat adaptar l'estil del còmic per a videojocs. Els més coneguts, son títols com, Borderlands de l'estudi Gearbox Software o The Walking Dead de Telltale. Aquests dos casos en concret es centren més en adaptar l'estil clàssic del còmic. A més no utilitzen la tècnica del cel shading, cosa que provoca que el resultat final sigui més realista i per tant s'allunya de l'estil que volem recrear. Tot i això, fan servir mètodes per a generar el outline de la geometria mitjançant textures amb els que s'obtenen bons resultats, que poden ser d'utilitat per a aquest projecte.

Per últim, respecte a la animació, l'estudi Arc System Works creadors dels videojocs Guilty Gear Xrv i Dragon Ball FighterZ, han aconseguit recrear molt bé com seria adaptar la animació 3D per a que doni la sensació de ser animació tradicional fent servir els mètodes explicats en l'apartat anterior.

3. Gestió del projecte

3.1 Procediment i Eines per al seguiment del projecte

3.1.1 GANTT



T1 - GANTT del projecte

3.1.2 Trello

https://trello.com/b/fyj2jolC/tfg

3.1.3 Github

https://github.com/mizquierdo97/ComicRendering

3.2. DAFO

Fortaleses	Debilitats
- Coneixement elevat de la programació gráfica, shaders... i el seu funcionament. - Experiencia treballant amb Unity. - Oportunitat de consultar o treballar directament amb l'il·lustrador en que es basa aquest treball.	- Poca experiència amb la pipeline de render de Unity. - Necessitat d'art extern o produir art per mi mateix. - Temps limitat per al desenvolupament.
Oportunitats	**Amenaces**
- Actualment existeixen pocs videojocs amb un estil semblant al que es vol aconseguir. - L'estil triat, és un estil que en general agrada molt a la gent.	- Tot i que son pocs, existeixen jocs que compten amb un estil semblant al que es vol aconseguir. - Poca documentació dels mètodes que s'han de seguir i del funcionament de la pipeline de render de Unity.

T2 - DAFO

3.3. Riscos i pla de contingències

Per tal de planificar millor el projecte, és interessant identificar quan abans millor quins son els riscos que es prendran, per tal de poder trobar una solució i en cas necessari poder reconduir el projecte a temps.

Limitacions del motor

Al utilitzar software de tercers per a fer el projecte, en aquest cas Unity. Cap la posibilitat que existeixin certes limitacions que m'impedeixin avançar de manera parcial o total algunes característiques del projecte. En cas de que aparegui alguna limitació tècnica per part de Unity, es poden proposar diverses solucions. Per exemple, en el cas de trobar limitacions en la pipeline de render, es podria optar per modificarla. Unity compta amb varies render pipelines. La primera es la que et dona per defecte, que és la més fàcil de treballar amb ella i la més fàcil d'entendre. Després et dona dos pipelines optimitzades per a dispositius mòbils i per a gràfics realistes. Per últim, et dona la opción de programar desde zero la teva propia pipeline. Aquesta última, tot i que et dóna molta llibertat a l'hora de treballar, requereix de coneixements elevat i de dedicar-li molt temps a que funcioni correctament.

L'altre opció en cas que apareguin limitacions per part de Unity, és retallar contingut. Ja que és possible que per a algunes funcions no valgui la pena dedicar-li excessiu temps a canviar de pipeline.

Limitacions en l'art

Tot i que aquest treball està centrat en la programació gràfica, necessita d'una gran càrrega d'art per a aconseguir el resultat desitjat. Aquesta càrrega d'art pot significar fer part de l'art de manera autònoma, o encarregar a que ho facin altres persones. En el cas d'encarregar-me jo de l'art ajudaria a iterar de manera més ràpida en cas de trobar algun problema, però també augmentaria la càrrega de treball.

En cas de no poder assolir el nivell d'art necessari, tant per falta de temps com de recursos, es limitaria el número d'escenes finals o en cas extrem, es realitzarien aquestes escenes amb geometria basica.

Animacions

Com ja s'ha comentat, les animacions poden ser difícils de realitzar ja que primerament l'art original en el que es basa aquest treball son únicament il·lustracions, per tant no existeix cap referència de com haurien de ser les animacions per tal que s'adaptessin a aquest estil. Per altre banda, si es volgués seguir els processos que han fet servir altres videojocs i simular una animació tradicional, com ha fet el videojoc guilty gear, es requeriria d'una gran càrrega de treball dedicada a animar els diferents personatges.

En cas de veure que se li hauria de dedicar una gran quantitat de temps a les animacions o simplement pel fet de veure que el resultat final no compensa, s'optaria

per a fer unes animacions més bàsiques, o en un cas extrem, eliminar els personatges eliminant aixi la necessitat de realitzar animacions.

Performance

Al tractar-se d'un videojoc, una de les coses mes importants son els FPS, ja que han d'anar a un mínim de 30. En general en els videojocs, el render d'aquest, és lo que més consumeix i lo que més et limita a la hora de treballar. Per això s'haura de tenir especial focus a la hora de treballar, intentant que els processos siguin optims i no consumeixin gaire.

Per tal d'aconseguir el minim de 30 FPS, s'haurà d'optimitzar la pipeline de render en el moment en que es comenci a notar un mal rendiment. També s'haura de tenir en compte que no tots els ordinadors tenen les mateixes característiques, això comportarà haver de provar-ho en diferents ordinadors per a asegurar de que funcioni correctament.

Si les baixades de FPS son molt significatives, és probable que s'hagin de retallar o eliminar funcionalitats per tal d'assegurar aquest mínim, encara que això signifiqui una baixada de la qualitat gràfica.

3.4. Anàlisi inicial de costos

Per a l'anàlisis de costos s'ha tingut en compte que la duració del projecte, seria de 6 mesos.

Mes	Preu Unitat	Unitats	Nº Unitats	Temps d'amortització (Mesos)	Mesos de utilització	Total
Programador	1.300,00 €	Mesos	6	-		7.800,00 €
Artista	1.300,00 €	Mesos	6	-		7.800,00 €
Ordinadors	1.000,00 €	Items	2	36	6	333,33 €
Pantalles	100,00 €	Items	4	36	6	66,67 €
Escriptoris	100,00 €	Items	2	60	6	20,00 €
Cadires	80,00 €	Items	2	60	6	16,00 €
Teclats i Ratolins	40,00 €	Items	2	24	6	20,00 €
Llicència Maya	1.984,00 €	Any	1	-		1.984,00 €
					Total	18.040,00 €

T3 - Análisis inicial de costos

4. Metodologia

El projecte consistirà en tres fases, preproduccio, producció i postproducció.

Preproducció:

Es farà un anàlisis inicial de l'estil artístic que es vol crear, analitzant les diferents característiques tals com color, il·luminació, outline, efectes...

Un cop es tingui clar les característiques de l'estil artístic, es procedirà a buscar documentació de com es poden aplicar per a videojocs. Quins son els mètodes que es fan servir i quins son els pros i contres, per tal d'escollir quin mètode s'adapta més a les necessitats del projecte.

Després d'aquesta primera fase de documentació, s'implementarà una primera iteració dels mètodes escollits en una escena de prova en Unity, per a tenir una primera idea de com pot ser el resultat final i de detectar possibles riscos amb marge de temps.

Producció:

Després d'haver fet una primera iteració i de tenir una escena bàsica funcional, es començarà a treballar en els mètodes triat per tal de millorar-los tant per a que el resultat gràfic sigui millor com per a que el rendiment d'aquest mètodes sigui òptim.

En aquesta fase, també es començarà a treballar en l'art, creant els models 3D dels escenaris i dels personatges i les textures d'aquests. Això pot significar haver de modificar alguns dels mètodes que es feien servir per a que s'adaptin a la nova geometria. Per últim es treballarà en les animacions dels personatges.

Al finalitzar aquesta fase del desenvolupament s'haurien de tenir dues escenes completes. Aquestes escenes serán una replica de dues de les il·lustracions originals de l'autor.

Postproducció:

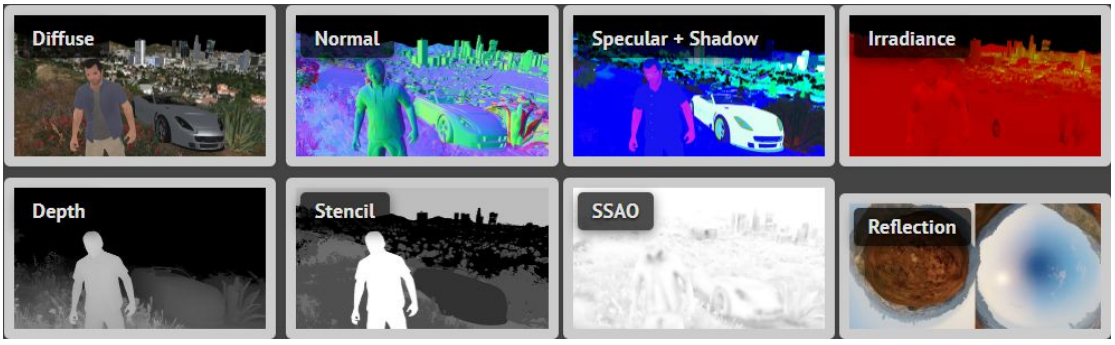
En la última fase del projecte, es desenvoluparà tota la documentació sobre el que s'ha après per tal que altres persones puguin replicar aquest estil o un altre de semblant. L'objectiu és tenir una documentació en anglès, que expliqui tots els mètodes que s'han seguit, com s'han aplicat en el projecte i els problemes que s'han anat trobant duran el desenvolupament del projecte.

Durant aquesta fase tambe es podran arreglar petits errors gràfics o de programació que es vagin trobant, pero sense introduir funcionalitats noves.

5. Desenvolupament del projecte

En aquest apartat s'analitzaran les diferents tecnologies que s'han fet servir en el desenvolupament d'aquest projecte i s'explicaran els mètodes que s'han seguit per a aconseguir el resultat final.

Camera Render Targets:
En la indústria dels videojocs i més concretament en la programació gràfica, de forma general es fa servir un mètode de render que permet al desenvolupador tenir un major control a l'hora de crear els gràfics del videojoc. Aquest mètode consisteix en què a l'hora de renderitzar l'escena, es generen vàries textures, amb diferents valors que poden ser útils per al posterior tractament dels gràfics. Aquestes textures poden contenir diferents valors com el color, la profunditat, la il·luminació...



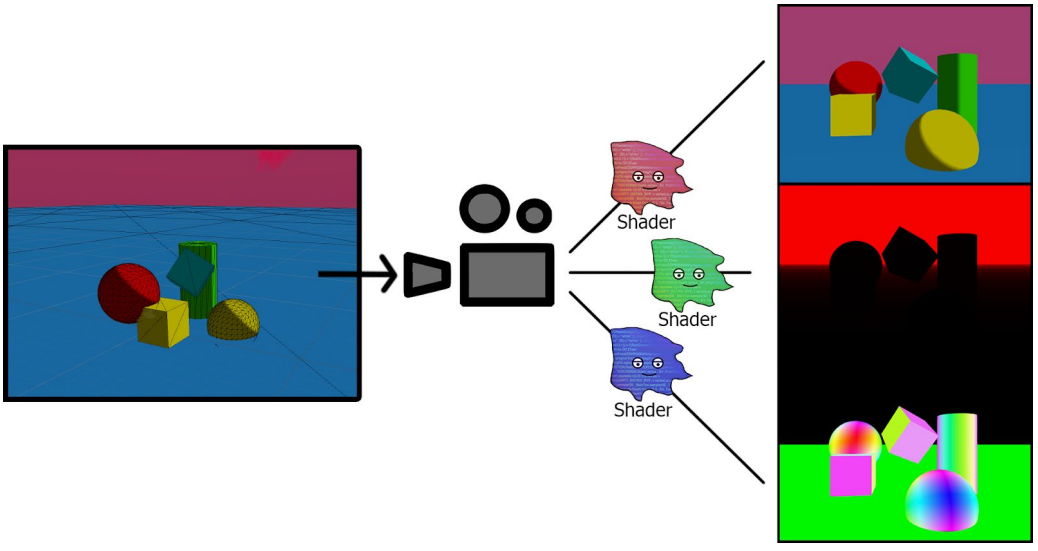
F8 - Render Targets del videojoc GTA V

Aquestes, es combinaran posteriorment fent servir diversos mètodes per a obtenir el resultat final desitjat. Aquest mètode es coneix com a deferred rendering, i és utilitzat sobretot per a aconseguir gràfics realistes, ja que permet obtenir una aproximació dels valors físics dels objectes per separat amb càlculs molt senzills per a després combinar-los.

F9 - Fotograma final del videojoc GTA V

En programació a més baix nivell, com per exemple OpenGL, és relativament fàcil implementar i modificar aquest mètode perquè s'adapti a les necessitats del projecte, ja que permet modificar quines textures vols obtenir a partir d'un únic pas de render de la geometria. És a dir, es renderitza la geometria una única vegada i d'aquesta geometria obtens tots els valors que vulguis en forma de textura.

Unity en canvi, no et deixa obtenir diferents textures en una única passada de la geometria i has de fer diverses passades aplicant un shader per a obtenir les textures desitjades. Això comporta un increment del temps d'execució que pot afectar en el rendiment del joc.



F10 - Esquema de la pipeline de render utilitzada

Per a aquest projecte, s'han fet servir 4 passades de render addicionals per tal d'obtenir les textures de profunditat i normals dels personatges i de l'escenari.

A més hi ha una passada addicional per a l'efecte de glow, en la que s'han utilitzat Assets externs per a aconseguir-ho.

Les textures creades (Render Textures) tenen una mida proporcional a la finestra en què s'estigui executant el joc (width, height). Com més grans siguin les textures més detall i menys errors gràfics apareixeran, però també comportarà un increment de la memòria utilitzada i del temps d'execució, ja que tindrà més píxels als que aplicar els filtres de postprocessat posteriors.

A més, s'han utilitzat textures amb una profunditat de 32 bits, a excepció de les normals dels personatges, ja que a l'augmentar aquest valor, ajuda en el posterior pas de detecció d'arestes al crear imatges més precises. Per últim en les textures de profunditat s'ha utilitzat només un canal, ja que només s'ha de tenir en compte un valor. D'aquesta manera, obtenim molta més precisió però sense consumir més memòria.

```
charactersNormals = new RenderTexture(width, height, 16, RenderTextureFormat.ARGBFloat);
charactersDepth = new RenderTexture(width, height, 32, RenderTextureFormat.RFloat);
objectNormals = new RenderTexture(width, height, 32, RenderTextureFormat.ARGBFloat);
objectDepth = new RenderTexture(width, height, 32, RenderTextureFormat.RFloat);
```

F11 - Funcions per a crear Render Target en Unity

Un cop creades les textures, en cada frame es fan les següents crides per tal que la càmera renderitzi un altra vegada l'escena però utilitzant el shader desitjat.

```
//Render Map Normals
CameraSetup(objectNormals, LayerMask.GetMask("Map"));
CharactersCamera.RenderWithShader(MapNormals, "RenderType");

//Render Map Depth
CameraSetup(objectDepth, LayerMask.GetMask("Map"));
CharactersCamera.RenderWithShader(MapDepth, "RenderType");

//Render Characters Normals
CameraSetup(charactersNormals, LayerMask.GetMask("Characters"));
CharactersCamera.RenderWithShader(CharactersNormals, "RenderType");

//Render Characters Depth
CameraSetup(charactersDepth, LayerMask.GetMask("Characters"));
CharactersCamera.RenderWithShader(CharactersDepth, "RenderType");
```

F12 - Funcions per a renderitzar l'escena utilitzant diferents Shaders

Edge detection:

Una de les característiques més importants de l'estil artístic del Josan, és el traç a l'hora de dibuixar. El procés que ell segueix, consisteix a fer el dibuix en paper per a posteriorment passar-ho a digital. En tractar-se d'un traç fet a mà fa que sigui bastant difícil replicar-ho en 3D fent servir algun tipus d'algoritme. És per això que la majoria de recursos s'han destinat a què el traç fos el semblant a l'estil original.

F13 - *Detall del traç de l'art original.*

El primer pas va ser investigar diferents algoritmes de detecció d'arestes, per tal de veure quin s'acostava més al resultat desitjat. En general, hi ha dos algoritmes principals, el Sobel i el Canny, la resta d'algoritmes, són adaptacions o deriven d'un d'aquests dos. De fet, el mateix Canny és una extensió del Sobel.

F14 - Resultats dels algoritmes Sobel i Canny

A primera vista, el Canny sembla que proporciona un resultat millor, ja que les arestes que genera són més clares, a més que aquestes tenen una amplada fixa d'un pixel. Per contra, l'algoritme Sobel, genera unes línies més irregulars, d'amplada variables, a més de generar un cert soroll on no haurien d'aparèixer línies. El problema amb el Canny, es que no es pot calcular en temps real, ja que fa servir uns càlculs per a descartar falses arestes bastant costosos, on es té en compte si les arestes amb poc pes estan o no connectades a arestes amb més pes. Si no ho estan es descarten.

Després de provar diferents aproximacions del Canny perquè es pogués fer servir a temps real, el vaig descartar, ja que no donava els resultats esperats. Finalment vaig adaptar l'algoritme Sobel aplicant-li un llindar per tal de descartar el possible soroll que es generés.

F15 - Algoritme Sobel aplicant-li un llindar de 0.8

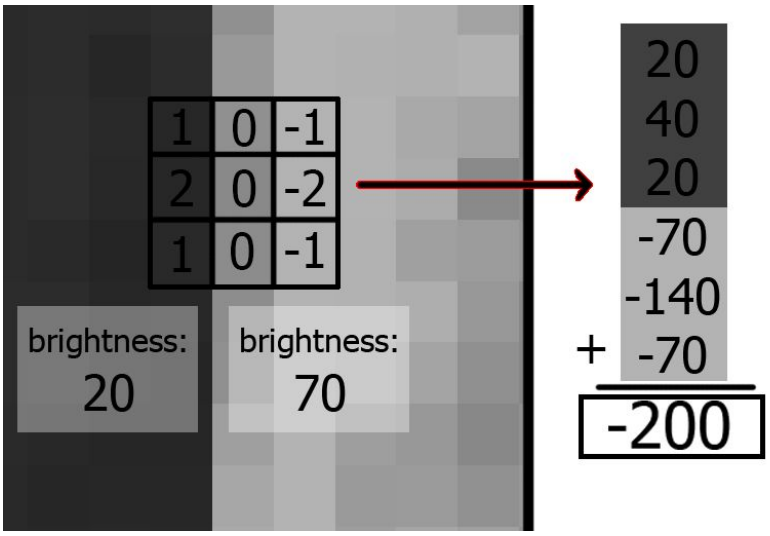
Aquest mètode possiblement no seria vàlid si el que tractessim d'obtenir fossin les arestes d'una sola imatge, ja que les arestes que no passessin el llindar, desapareixerien. Però al tenir vàries imatges sobre les quals detectar arestes com el color, la profunditat i les normals, és bastant difícil que es descarti una aresta, ja que si per exemple en la textura de profunditat no se'n detecta una, és molt probable que sí que es detecti en la textura de normals.

Sobel:

El funcionament de l'algoritme Sobel es relativament senzill. Primer de tot es necessita la textura a la qual es vulguin detectar les arestes, en el nostre cas tres textures, color, normals i profunditat. Seguidament es farà un análisis de cada pixel utilitzant dues matrius, una per trobar canvis de gradient en vertical i un altre pels canvis en horitzontal. Aquestes matrius analitzaran els píxels del voltant del qual s'està analitzant i determinaran el canvi de gradient.

Per exemple en el cas de la matriu horitzontal, es multipliquen per un valor positiu els píxels de l'esquerra i per un valor negatiu els de la dreta, de manera que al final obtens un resultat. Si s'ha trobat un gradient, el resultat serà diferent de 0. En el cas que el resultat sigui 0, voldrà dir que no hi ha una variació horitzontal, ja que a l'aplicar l'operació els valors dels dos costats en contrarestaran.

El signe del resultat, si és positiu o negatiu, indicarà la direcció del gradient, però en el nostre cas és irrellevant i no es tindrà en compte.

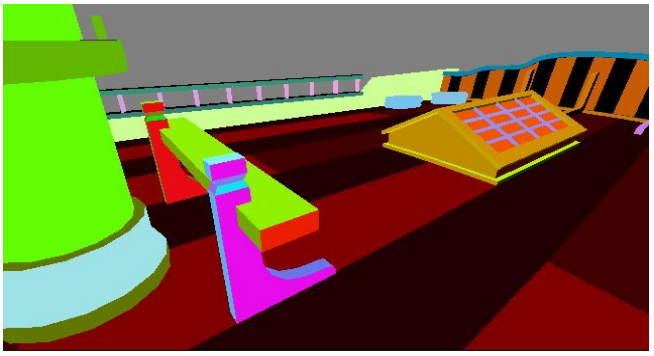


F16 - Funcionament del algoritme Sobel

Amb aquest mètode podem començar a dibuixar un outline als objectes, però encara presenta alguns errors gràfics i dista del resultat que es vol aconseguir, ja que no dóna la sensació de ser un traç fet a mà. Per aconseguir-ho, i mitjançant la textura obtinguda amb l'algoritme Sobel, aplicarem diferents tècniques i filtres de postprocessat per a arreglar els errors gràfics i millorar el outline.

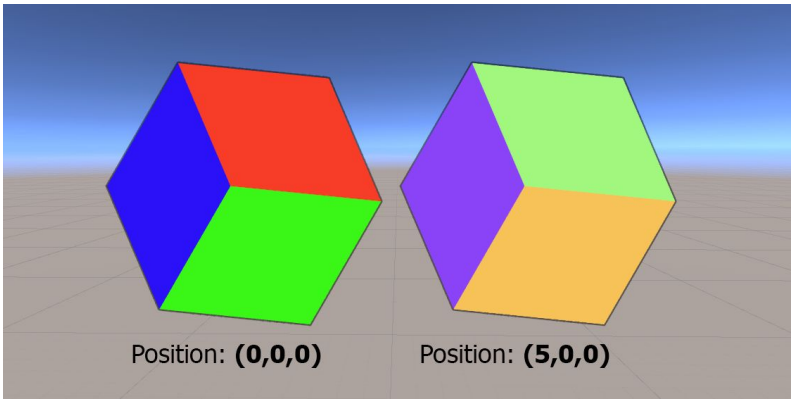
Normals dels objectes:

Un dels motius principals pels quals no es dibuixen les arestes de certs objectes, és que tant en la textura de normals com en la de profunditat, no hi ha una diferència significativa per a detectar una aresta, encara que es tracti de dos objectes diferents. Després de buscar possibles solucions per a aquest problema, la que em va semblar més òptima és la que utilitza el videojoc Return of The Obra Dinn.



F17 - Textura de normals del videojoc Return of the Obra Dinn

El mètode que utilitza aquest videojoc, és el de renderitzar les normals, però tenint en compte no només la pròpia direcció de la normal de cada vèrtex, sinó la posició de cada objecte. Consisteix a aplicar una rotació a totes les normals de l'objecte en qüestió, de manera que dos objectes amb la mateixa rotació però diferent posició, es renderitzaran en la textura de normals amb color diferent, ja que internament aquestes normals estaran "rotades" en funció de la posició.

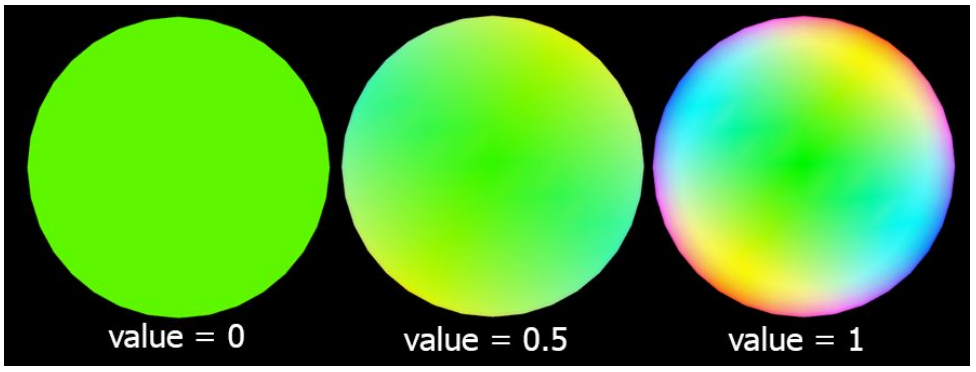


F18 - Normals d'un objecte depenent de la seva posició

Tot i que aquest algoritme arregla la gran majoria dels errors gràfics, es pot donar la casualitat que dos objectes es renderitzin amb el mateix color i que per tant no es

dibuixin les arestes. Això es pot solucionar fàcilment aplicant un offset a mà en els objectes que donin aquest problema.

Un altre dels errors gràfics que es presentava, era en els objectes més esfèrics, ja que en la seva superfície hi apareixen variacions en les normals però no s'hi haurien de dibuixar arestes, ja que es tracta de superfícies corbades. Igual que amb l'error gràfic anterior, adaptarem la solució que fa servir el videojoc Return of the Obra Dinn. La solució que ells proposen consisteix en què cada material té un valor que determina la importància de les normals per a cada objecte. Fent que per a certs objectes s'utilitzi els valors de les normals obtinguts amb el mètode anterior, o que per contra, s'utilitzi simplement la posició de l'objecte Ignorant les normals. Per posar un exemple, així és com es renderitzaria una esfera en funció d'aquest valor.

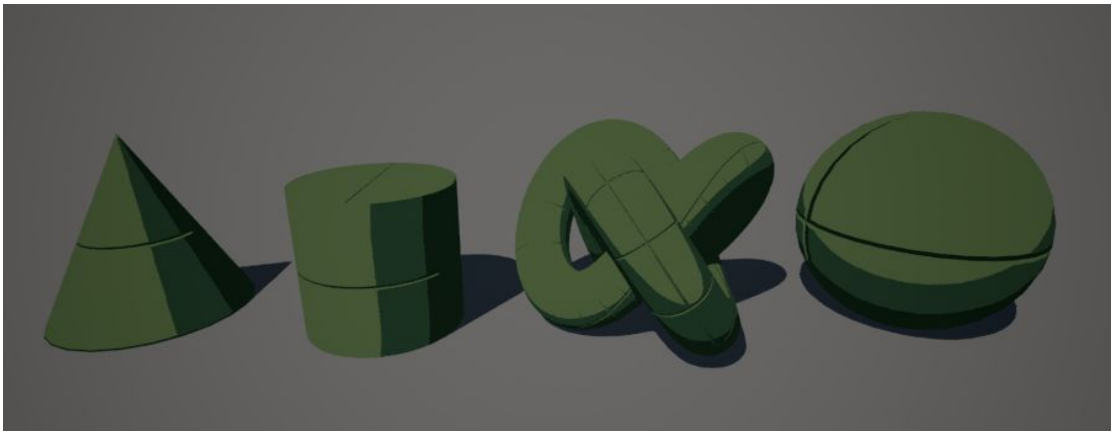


F19 - Render de les normals per a diferents valors

Cel Shading

Per a aconseguir un estil més cartoon, en la indústria dels videojocs es fa servir de manera general una tècnica coneguda com a Cel Shading. Aquesta tècnica es fa servir per a gràfics no-fotorealistes i consisteix principalment a aplicar la il·luminació en els objectes de manera discreta en comptes de continua, creant l'efecte d'una il·luminació plana que fa que s'assembli més a l'estil Cartoon.

Aquesta tècnica s'ha utilitzat en diversos videojocs. Des de el primer que ho va fer servir, Jet Set Radio l'any 2000, passant per The Legend of Zelda: Wind Waker o Dragon Ball FighterZ entre d'altres.

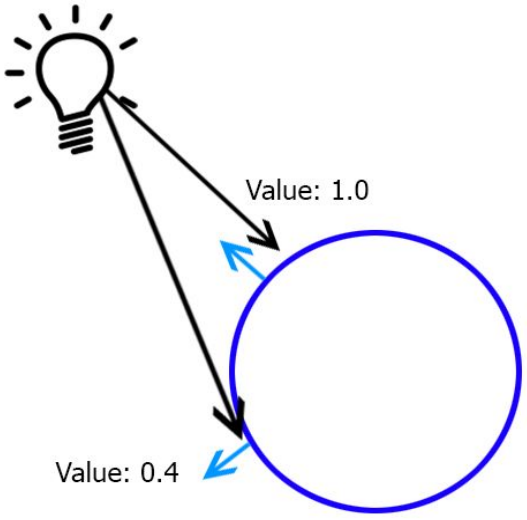


F20 - Exemple de Cel Shading

El procediment per a aconseguir aquest resultat, és relativament senzill.

A l'hora d'il·luminar un objecte, depenent de la direcció de la superfície de l'objecte i de la direcció de la llum es fa el producte escalar d'aquest dos vectors, que retornarà un valor entre 1, si la llum li dóna directament, amb un angle de 0 graus, a -1 si els dos vectors són iguals i per tant no rep llum.

En la il·luminació convencional, simplement s'agafa aquest valor i se li aplica a l'objecte, creant una il·luminació continua. En canvi el cel shading el que fa és agafar un llindar a partir del qual els valors que estiguin per sobre d'aquest se'ls hi atorgarà un valor únic, i els que estiguin per sota d'aquest, se'ls hi donarà un altre valor, creant així una il·luminació plana.



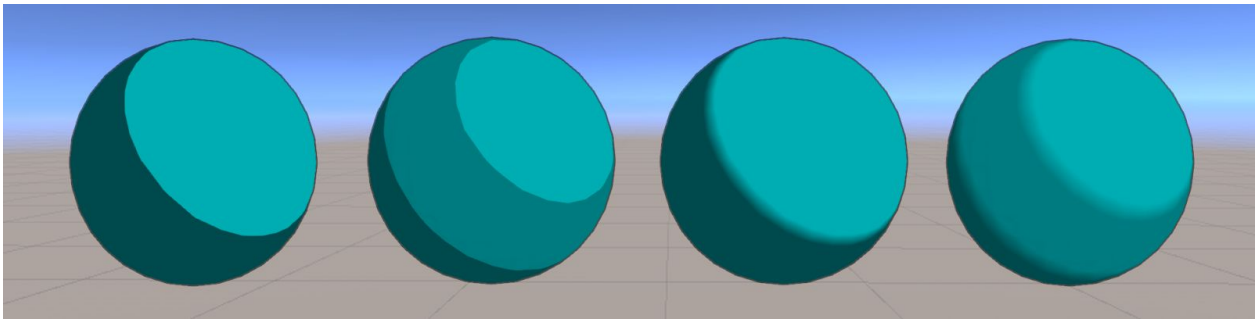
F21 - Exemple de valor depenent de la inclinació de la llum

Per exemple si agafem com a llindar el valor 0.5, tots els valors que estiguin per sobre d'aquests se'ls hi assignarà 1, en canvi, els que estiguin per sota 0.

Partint d'aquesta base, vaig fer alguns canvis perquè s'adaptés millor a l'estil artístic del Josan.

Un dels punts que personalment m'agrada menys d'aquest mètode és que crea una divisió molt marcada entre les parts il·luminades i les parts amb ombra, ja que en la meva opinió, aquesta divisió, fa que s'allunyi d'un dibuix fet a mà, on les dues parts queden més difuminades. Per aquest motiu vaig afegir una petita transició entre les dues parts.

A més vaig afegir un tercer llindar, per a les parts menys il·luminades creant així tres nivells amb diferents intensitats de la llum. A continuació es pot veure un exemple dels diferents resultats amb aquestes característiques.

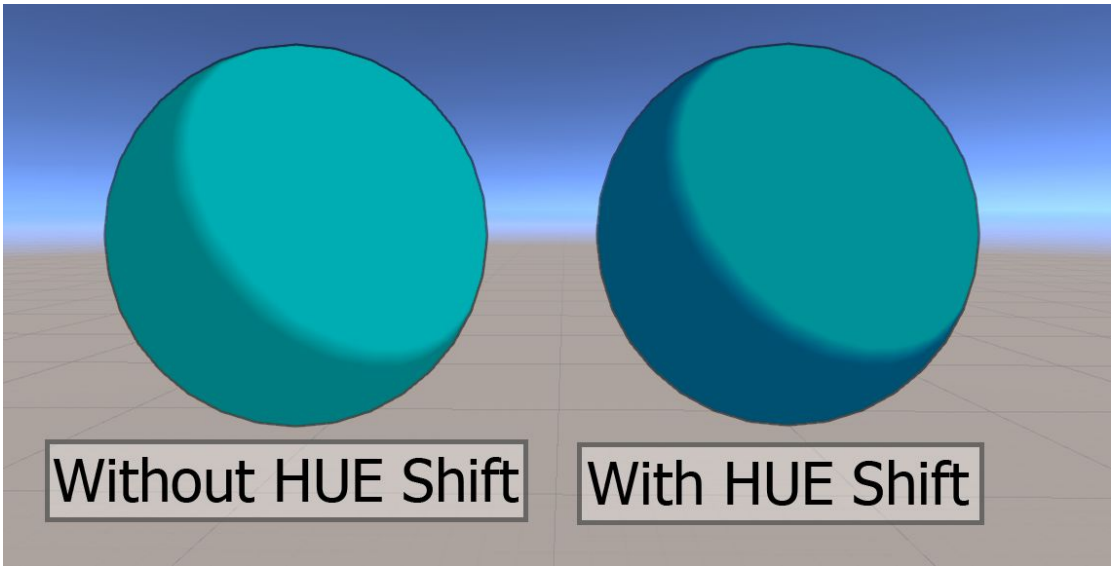


F22 - Diferents Cel Shading implementats en el projecte

Per últim, després d'analitzar els colors en l'art original, vaig veure que les ombres feien servir una variació del color base, de manera que les ombres tenien un color diferent de les parts il·luminades. De fet això és bastant comú en el ús dels colors en les il·lustracions, ja que si no es fa aquesta variació de colors, el resultat final queda molt pla i sense gaire interès.

Tot i això aquesta tècnica no se sol fer servir en videojocs, o no he pogut trobar casos específics on s'utilitzes, on únicament s'enfosqueix el color base però sense canviar-li la tonalitat.

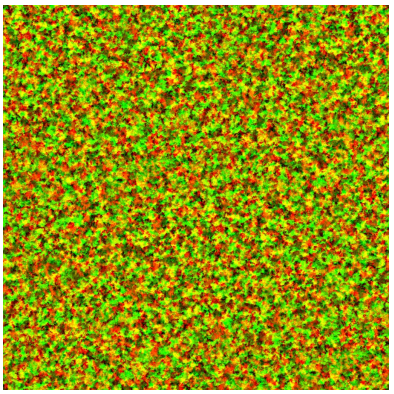
En l'exemple a continuació es pot observar aquesta variació, que tot i que és subtil, li dóna més interès a la imatge.



F23 - Diferencia sense o amb HUE Shift

Outline Postprocess:

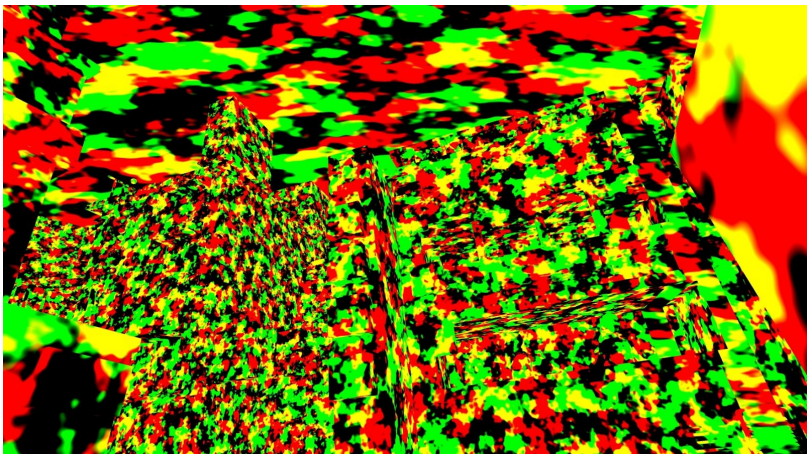
Per tal de dotar de més interès al outline i evitar que les línies generades fossin totalment rectes, fet que feia que és notes molt que es tractava de geometria 3D en comptes d'una il·lustració, he aplicat un parell de tècniques i filtres de postprocessat que ajuden a crear aquesta sensació.



F24 - Textura de soroll utilitzada per distorsionar les linies

El primer filtre consisteix a distorsionar les línies en els eixos X i Y, perquè doni la sensació de línies imperfectes fetes a mà. Per a aconseguir-ho, vaig començar utilitzant una textura de soroll, per a ajudar a que el procés semblés més aleatori.Si utilitzés un algoritme per a fer aquesta distorsió molt probablement es notaria un cert patró o repeticions en el resultat final.

Aquest mètode al utilitzar una textura en espai de pantalla, provocava que al moure la camara, les linies es moguessin i creava un efecte bastant extrany. Per a arreglar això, en comptes d'utilitzar una textura en espai de pantalla, he creat la textura de soroll mitjançant la propia geometria que es volia renderitzar. Aquesta textura es crea en cada frame, depenent del que estigui veient la camara en aquell moment i crea el soroll mitjançant l'input de posició global del pixel que es vol pintar.



F25 - Textura de soroll en funció de la posició global de la geometria

Com es pot veure, el soroll queda generat en funció de la posició de la geometria de la escena, el que provocarà que les línies que es pintin siguin estàtiques i no canviin depenent de la posició de la cámara.

Un cop es té aquesta textura, el procés per a distorsionar les linies consisteix bàsicament a utilitzar el canal R de la imatge per a moure els píxels en l'eix X i el canal G per a moure els píxels en l'eix Y. Aquest desplaçament depèn de la profunditat on es trobi el pixel, per tant les línies que es trobin més lluny, tindran una major distorsió. Això, a part d'ajudar a crear una certa sensació de profunditat i de donar-li més importància als objectes que estiguin més propers, és una tècnica que també s'utilitza en l'art original.

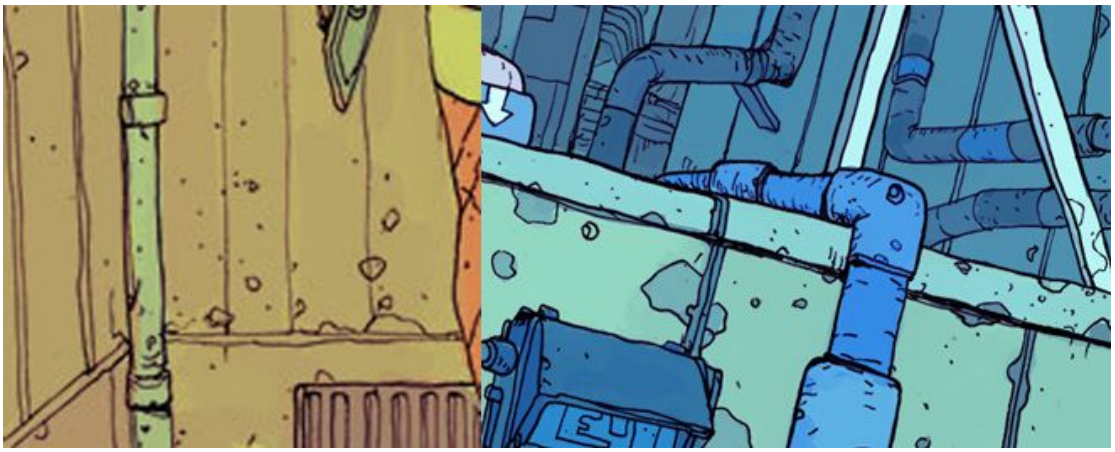
F26 - Detall de la distorsió amb la distància en l'art original

Aquest procediment també l'he implementat per la textura de color, per tal de crear una distorsió en els objectes més llunyans i crear l'efecte que s'ha "pintat" per fora de la línia.

A part d'aquest efecte, i utilitzant la mateixa textura de soroll que amb l'efecte anterior, vaig implementar un mètode semblant perquè les línies tinguessin certa variació amb la intensitat, fent que certes parts de les línies fossin més fosques i més gruixudes i altres, més clares i fines.

El procediment és molt semblant a l'anterior. Bàsicament s'utilitza el mateix canal R per a, en comptes de moure un pixel, canviar-li la intensitat. Per exemple si la línia en un cert pixel, té valor 1, però la textura en aquell punt té valor 0, al píxel final li donarem un valor de 0.5. El que es vol fer és crear irregularitats en la línia, no que desaparegui per complet, per això s'assigna 0,5 en comptes de 0 per a un valor de la textura de 0.

Textures:
A part del outline de la geometria, les imatges originals, també presenten un outline en els mateixos objectes que no depenen de la geometria. Aquest traç, no es pot crear fent servir les tècniques anteriors de postprocessat i per tant s'ha de buscar un mètode alternatiu per a crear-lo.



F27 - Detall de les textures en l'art original

La primera opció va ser la d'aplicar aquest outline directament en la textura de cada objecte. Això però, portava una sèrie de problemes.

El primer, era que s'havia de fer una textura única a mà per cada objecte. Si només es tractés d'una escena podria servir, però si es vol aplicar en un videojoc, aquest mètode seria bastant ineficient, ja que es necessitaria una gran quantitat de textures.

El segon problema era que aquestes textures tenen una resolució fixa i perden qualitat si t'acostes als objectes i que per poder veure un outline ben definit, aquestes textures haurien de tenir una resolució molt gran perquè a l'estar a prop no es perdi detall. Això òbviament comportaria un consum de memòria molt notori, que perjudicaria el rendiment del videojoc.

Com a punt a favor, aquest mètode et donava el control total sobre el outline, però pels punts negatius es va descartar aquesta tècnica.

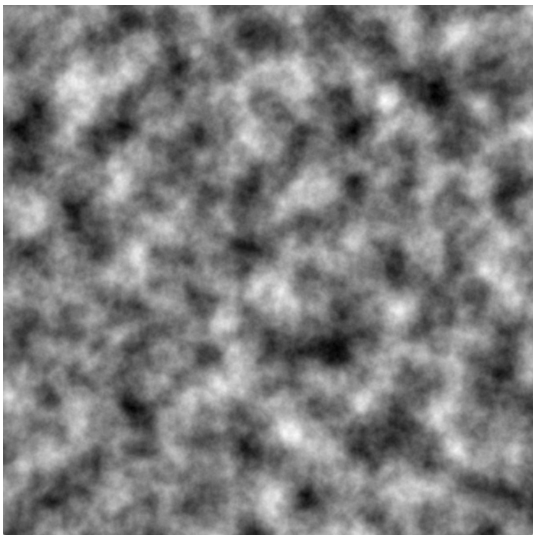
Com a alternativa, estava l'opció de fer aquesta textura de manera procedural mitjançant algoritmes de soroll procedural. Al principi tenia els meus dubtes de si aquest mètode podia funcionar correctament. Però al final, el resultat ha estat bastant satisfactori.

El primer pas per a aconseguir aquest efecte, era el d'obtenir textures de soroll de manera procedural. Això volia dir que es poguessin crear en temps d'execució i que depenguessin d'uns valors fàcilment modificables. Per sort, en la Store de Unity hi havia assets gratuïts que proporcionaven el codi per a crear aquestes textures en temps d'execució. De no existir aquests assets, possiblement hauria d'haver descartat

aquest mètode, ja que per a la creació d'aquests algoritmes es necessita haver-se documentat bastant i dedicar-li bastant temps per a implementar-lo.

Un cop tenia el codi que em permetia crear aquestes textures, vaig investigar una mica quins diferent algoritme em proporcionava i veure quin s'adaptava més a el que necessitava. Finalment, vaig escollir l'algoritme anomenat Perlin. Un algoritme molt conegut que es va dissenyar per a crear textures en la pel·lícula de Disney "Tron" de l'any 1982, i que va ser reconegut amb un Oscar.

L'algoritme Perlin consisteix en una sèrie d'interpolacions entre un gran nombre de gradients precalculats de vectors, que construeixen un valor que varia aleatòriament depenent de la posició del pixel. En definitiva, que et genera textures semblants a aquesta:



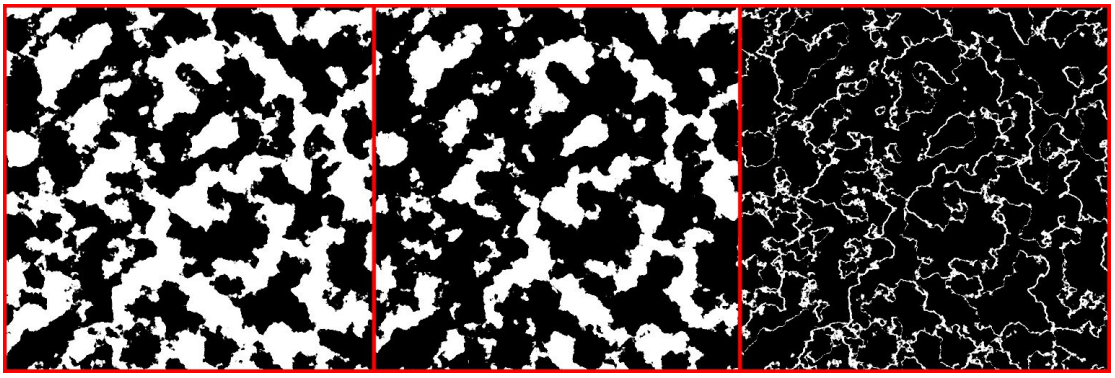
F28 - Resultat de l'algoritme de soroll Perlin

En el nostre cas, el soroll generat dependrà de la posició global del pixel que es vulgui pintar, de manera que cap objecte tingui una textura igual.

Tot i que tenim la textura del soroll, aquesta no s'assembla gens al resultat que volem aconseguir, però tenim una base sobre la qual començar a treballar.

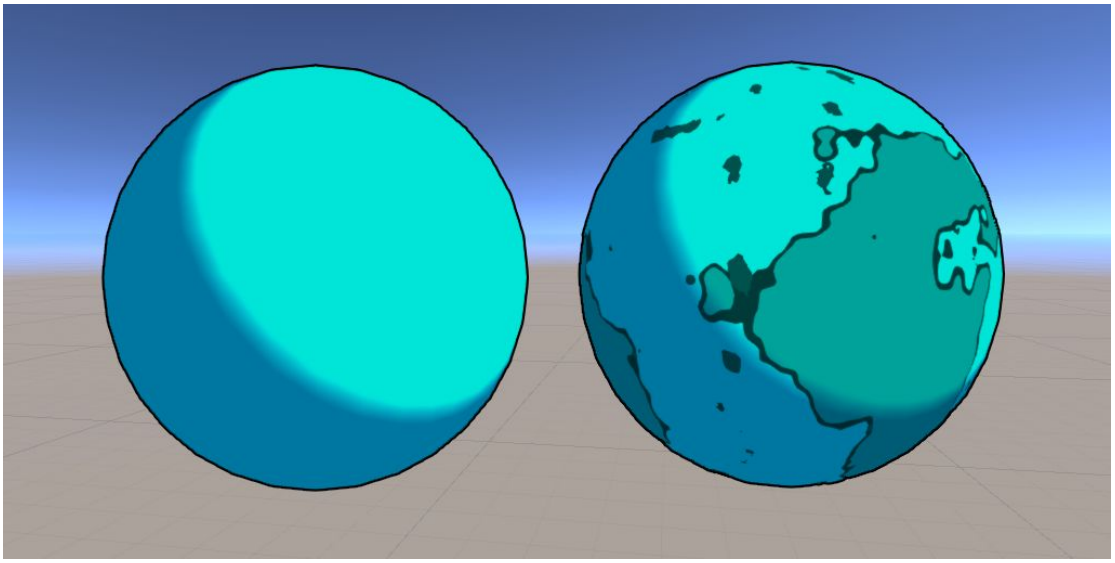
El primer que cal fer és obtenir una imatge en blanc i negre, sense tonalitats intermèdies, a partir de l'original. Si un valor és inferior a un cert llindar, el valor final serà 0, si és superior serà 1. Si fem el mateix pas, però amb un llindar lleugerament superior, obtindrem una imatge semblant, però on les àrees blanques seran lleugerament menors.

Restant aquestes dues textures obtenim el següent resultat:



F29 - Procediment per a obtenir les textures

Seguint aquest procediment, obtenim una textura que es comença a acostar al resultat que es volia. Si seguim jugant amb alguns valors de l'algoritme, com la freqüència del soroll o el llindar, obtenim un resultat més fidel. La diferencia de fer servir o no aquesta tècnica és bastant notable i fa que el resultat final s'assembli més a l'art original, tot i no ser perfecte.

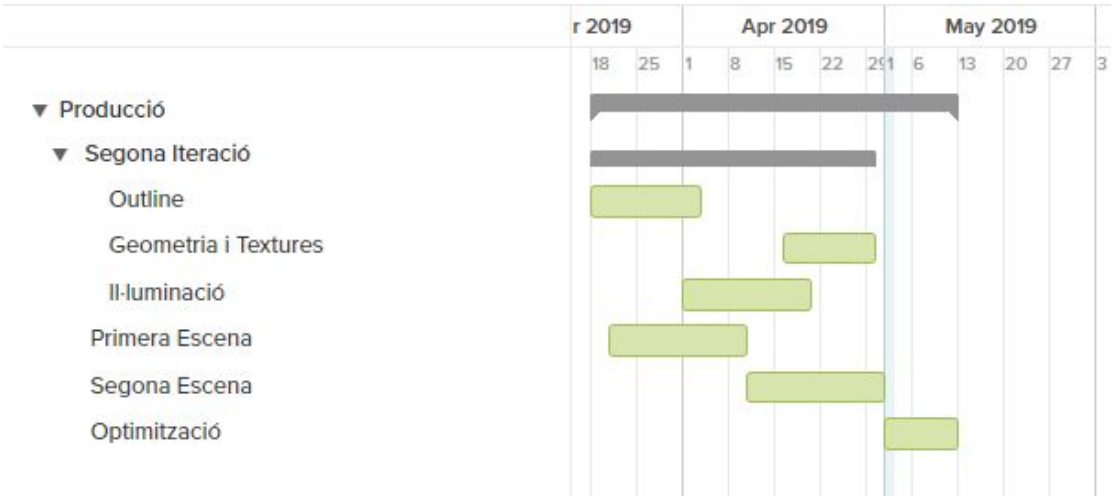


F30 - Resultat final aplicant-li les textures generades amb soroll

Glow i animació:
Per falta de temps, hi ha hagut certs aspectes inicialment previstos que s'han hagut de retallar en contingut o modificat. Aquests dos casos són l'efecte de glow i les animacions tradicionals.

En el cas del glow he fet servir un asset de la Store de Unity, ja que primerament em proporcionava el resultat que jo volia i, segon, m'estalviava temps que podia fer servir en altres característiques del treball.

Per altra banda, l'estil d'animació tradicional també ha estat descartat, ja que és una característica que comportaria bastant temps de treball a més que el personatge sobre el qual haig de treballar encara no està acabat. Per això, es farà servir un estil d'animació bàsic.



F31 - Variació en la planificació inicial

Iteració i finalització.

Al tractar-se d'un treball relacionat amb l'aspecte gràfic, gran part del temps ha estat dedicat a iterar i modificar certs aspectes del projecte, per a que quedés, en la meva opinió, lo millor possible. Això per exemple pot significar dedicar-li varies hores a veure quins valors quedaven millor per a la generació de soroll, quantes variacions de soroll diferents necessitava, si algun valor havia de dependre d'algun altre valor com per exemple la profunditat, per a minimitzar artefactes, etc...

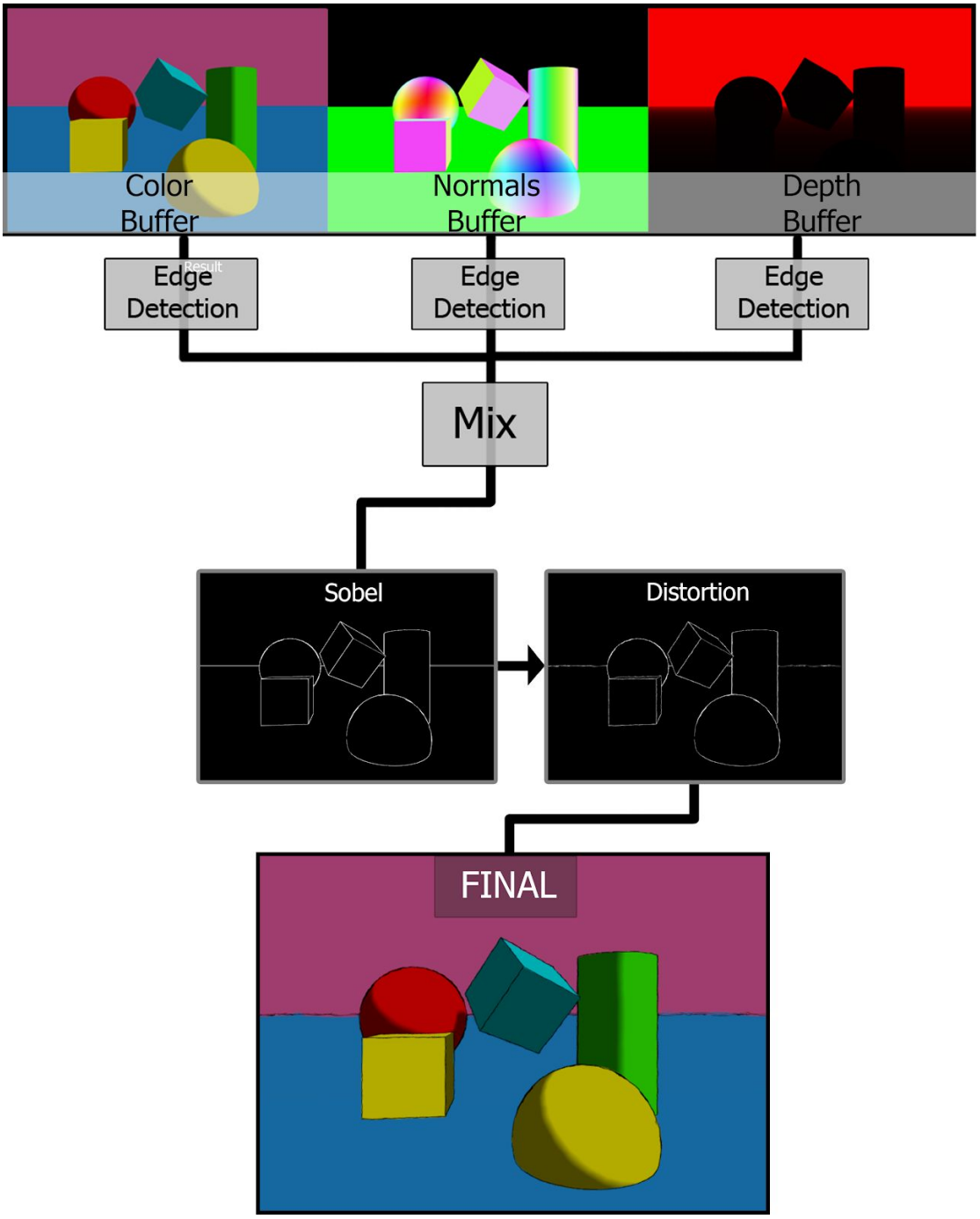
Aquesta iteració la he tingut que fer en cada mètode implementat i en certs casos, varies vegades per a alguns mètodes, ja que al modificar o afegir certs aspectes podia ser necessari tornar a revisar els valors.

A més també li he dedicat temps a provar d'afegir certs mètodes de render, com per exemple efectes de postprocessat com el SSAO, o altres efectes dintre del shader per a intentar simular variacions de color semblants a les dels dibuixos fets a mà.

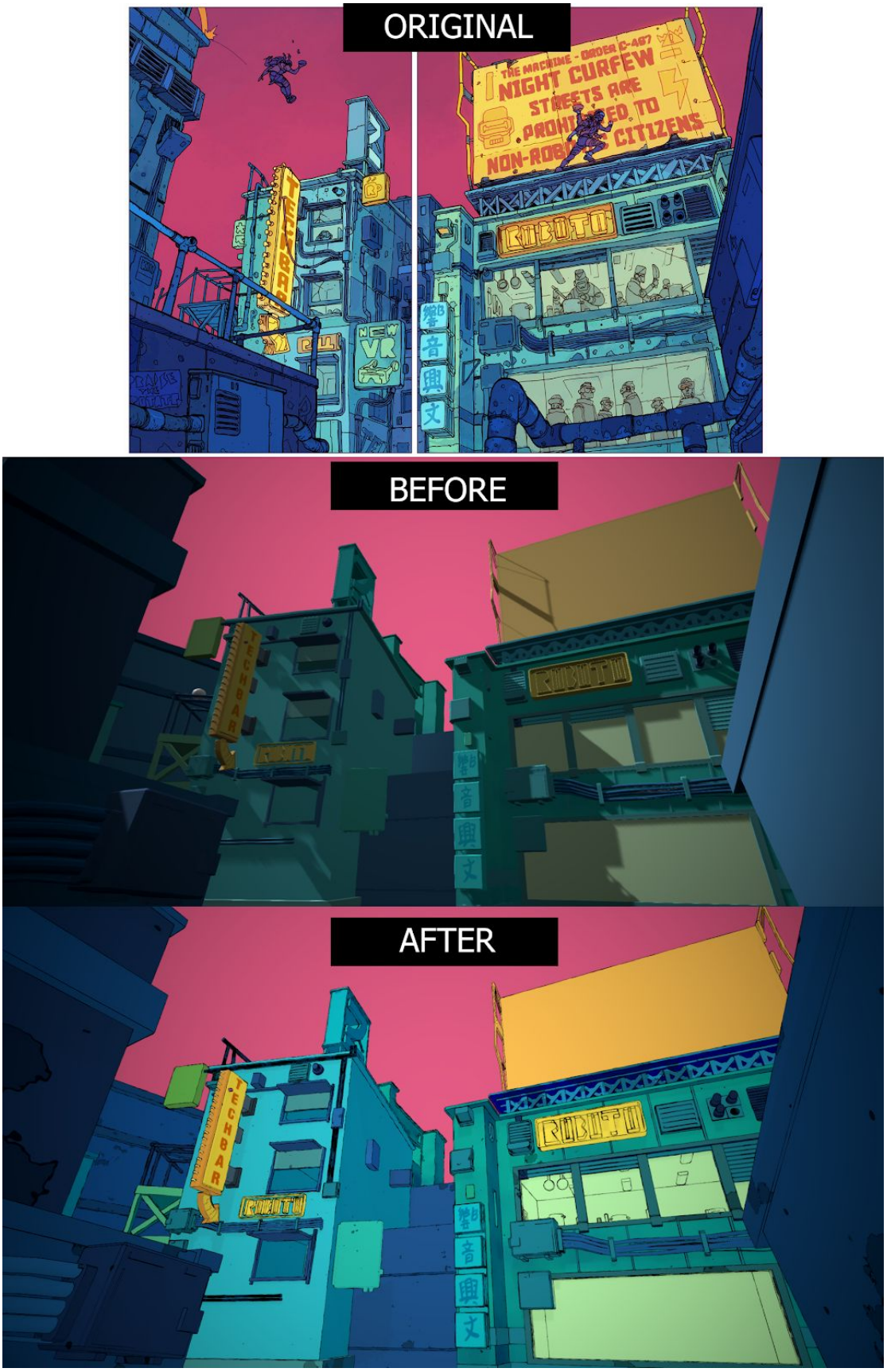
Aquesta iteració, tot i que era poc perceptible, ja que al final estaba modificant detalls del resultat final, també m'ha ocupat bastant temps, ja que he necessitat fer recerca, implementació i iteració de, tot i que senzills, bastants mètodes diferents.

Per últim, també li he tingut que dedicar bastant temps per a arreglar bugs i millorar la performance, que tot i que no era de gran importància per a aquest treball, m'ha impedit que pugui avançar en altres aspectes del projecte.

Per a aquest motiu, s'han tingut que retallar certs aspectes que al principi s'havien plantejat, com per exemple el glow, l'animació o diferents efectes de partícules per a centrar-me principalment en el render de geometría.



F32 - Esquema de la pipeline de render final



F33 - Art original, i resultat final dintre de Unity

6. Conclusions i treballs futurs

Com a conclusions, primerament m'agradaria fer un repàs dels objectius plantejats a l'inici del projecte per a veure quins s'han completat i com.

Estil Gràfic

L'objectiu principal d'aquest treball, era el de recrear l'estil artístic escollit, perquè s'assemblés el màxim possible a l'art original. Aquest objectiu crec que s'ha complert satisfactòriament, ja que el resultat final, tot i no ser idèntic, s'acosta bastant al que es volia aconseguir. També cal remarcar que el resultat final s'ha aconseguit sense necessitat d'un artista que faci models 3D o textures, per el que encara es podria haver aconseguit un resultat millor.

Pipeline de Unity

Un altre dels objectius, era el d'aprendre amb profunditat com funcionava la pipeline de render de Unity. A primera vista, aquest objectiu podia semblar bastant assequible, ja que entendre una de les característiques de Unity, no hauria de ser massa complicat. El problema que m'he trobat és que aquesta pipeline és molt més complexa del que pot semblar en un principi i té molts aspectes en els que no he pogut profunditzar. Tot i això, sí que he après molt bé altres tècniques de la pipeline, com les passades de render, el funcionament dels shaders o el tractament de la il·luminació entre d'altres. Per lo que, personalment crec que s'ha assolit aquest objectiu.

Documentació

A causa de la gran càrrega de treball que ha suposat assolir els altres objectius, he hagut de deixar com a treball futur la producció d'una documentació per a poder entendre i recrear el funcionament dels shaders i efectes treballats en aquest projecte.

També ha estat a causa que fins a les etapes finals del projecte, he estat treballant en algunes de les tècniques i característiques de la pipeline de render, per lo que tampoc em podia posar a redactar una documentació si existia la possibilitat de canviar el funcionament d'alguna de les característiques.

Mètodes usats en la indústria

L'últim objectiu general que em vaig proposar per a aquest treball, era el d'aprendre quines eren les tècniques que es feien servir en la indústria per a aconseguir diferents efectes. Gran part d'aquest objectiu, el vaig assolir en l'apartat de Estat de l'Art, ja que em vaig haver de documentar sobre diferents mètodes per a saber quins podia fer servir en un principi.

A més durant el desenvolupament del projecte, he tingut que investigar alguns mètodes amb més profunditat perquè s'adaptessin a les necessitats del treball.

Objectius Específics

Dels objectius específics plantejats, crec que s'han assolit la majoria, començant per l'outline, on s'ha aconseguit un resultat bastant satisfactori, igual que amb el color, la il·luminació i el rendiment, ja que el projecte aconsegueix anar a més de 60 FPS, que era el límit que m'havia plantejat al començament.

Per altra banda, les animacions de personatges i altres efectes com el glow, com ja he explicat en altres apartats, s'han hagut de descartar per falta de temps.

Conclusions

En quant a les conclusions finals del treball, crec que el resultat obtingut, és bastant satisfactori, i he assolit en gran part, les expectatives que tenia en un principi. Ja que he arribat a aproximar bastant bé, o almenys en la meva opinió, l'estil de les il·lustracions originals. A més, un altre dels objectius, era el d'aprendre com funcionava la pipeline de render de Unity. Aquest objectiu, també crec que l'he assolit, tot i que a causa de la gran complexitat de la pipeline de render del motor, m'he deixat moltes coses que tant per falta de temps, com perquè el projecte no ho requeria, no he profunditzat en elles. Alguns d'aquests aspectes són el funcionament de la "High Definition Render Pipeline" per a simular gràfics realistes, com funciona el deferred render en Unity o programar la meva propia pipeline des de zero, entre d'altres.

Per altra banda, m'hauria agradat assolir tots els objectius proposats inicialment, però tant per falta de temps, com per una mala planificació inicial al principi del projecte i de voler assolir masses objectius, no he tingut temps per a fer-ho tot.

Per sort, aquesta falta de temps la vaig veure amb suficient temps per a reaccionar, cosa que em va permetre centrar-me en l'objectiu principal del projecte, i poder treballar en ell perquè el resultat final d'aquest objectiu fos el millor possible, en comptes de tenir tots els objectius assolits, però amb un resultat inacabat.

En conclusió, tot i no haver assolit alguns dels objectius proposats inicialment, crec que l'objectiu principal de recrear l'estil artístic, l'he assolit satisfactòriament. Espero que els coneixements apresos en aquest treball, em serveixin de cara al futur, i poder utilitzar aquest estil artístic per al desenvolupament d'un videojoc, ja que personalment, crec que té un potencial increïble, per a fer-ne un videojoc.

7. Bibliografia

Josan Gonzalez - ArtStation Josan Gonzalez (URL). https://www.artstation.com/josan [Consulta 14 Gener 2019]

Leafstormblaze - Comic Style Test Unity Engine (URL).

https://www.youtube.com/watch?v=-rvgRuEMo2E [Consulta 14 Gener 2019]

Raw Fury - Sable - E3 2018 Announcement Trailer (URL).
https://www.youtube.com/watch?v=KJFQhEY9udE [Consulta 18 Gener 2019]

GDC - GuiltyGearXrd's Art Style : The X Factor Between 2D and 3D (URL).
https://www.youtube.com/watch?v=yhGjCzxJV3E [Consulta 21 Gener 2019]

Wikipedia - Edge Detection (URL). https://en.wikipedia.org/wiki/Edge_detection [Consulta 22 Gener 2019]

Computerphile - Finding the Edges (Sobel Operator) (URL).
https://www.youtube.com/watch?v=uihBwtPIBxM [Consulta 22 Gener 2019]

UnrealCG - Outline Transition Effect Using Postprocess Material (URL).
https://www.youtube.com/watch?v=-7DXS7X3nn8 [Consulta 30 Gener 2019]

Unity Technologies - Edge Detection via Shader not Image Effect (URL).
https://forum.unity.com/threads/edge-detection-via-shader-not-image-effect.368922
[Consulta 5 febrer 2019]

William Chyr - Orthographic Edge Detection (URL) http://williamchyr.com/tag/edge-detection/
[Consulta 6 febrer 2019]

TIG Forums - Return of the Obre Dinn (URL)
https://forums.tigsource.com/index.php?topic=40832.msg1027183#msg1027183 [Consulta 14 febrer 2019]

Zehn Games - Next gen Cel Shading con Unity 5 (URL).
http://www.zehngames.com/developers/next-gen-cel-shading-con-unity-5/ [Consulta 17 febrer 2019]

Unity Technologies - Surface Shader lighting examples (URL)

https://docs.unity3d.com/Manual/SL-SurfaceShaderLightingExamples.html [Consulta 17 febrer 2019]

Wikipedia - Perlin Noise (URL). https://en.wikipedia.org/wiki/Perlin_noise [Consula 2 Març 2019]

Github - Noise Shader (URL) https://github.com/keijiro/NoiseShader [Consulta 3 Març 2019]

TheBookOfShaders - Noise Algorithms (URL)https://thebookofshaders.com/11/

[Consulta 5 Març 2019]

Unity Technologies - Rainbow Hue Shift (URL)

https://forum.unity.com/threads/solved-rainbow-hue-shift-over-time-c-script.351751/

[Consulta 10 Març 2019]

StackOverflow - Change Saturation (URL)

https://stackoverflow.com/questions/13806483/increase-or-decrease-color-saturation

[Consulta 10 Març 2019]

Unity Technologies - Using Projection Matrix to create holographic Effect (URL).

https://forum.unity.com/threads/using-projection-matrix-to-create-holographic-effect.291123/

[Consulta 28 Abril 2019]

Unity Technologies - WebGL Building (URL)
https://docs.unity3d.com/Manual/webgl-building.html [Consulta 10 Juny 2019]